Section IV:
AMENDMENT UNDER 37 CFR §1.121
REMARKS

Rejections under 35 U.S.C. §103

In the Office Action, claims 1 - 19 were rejected under 35 U.S.C. §103(a) as being unpatentable over US Published Patent Application 2002/0147857 to Sanchez, II, et al. (hereinafter "Sanchez") in view of US Published Patent Application 2005/0021498 to Boreham (hereinafter "Boreham").

Rejection of Independent System Claim 1. In rejecting Claim 1 over Sanchez in view of Boreham, the Examiner reasoned:

> *Examiner:*
> "Referring to claim 1, Sanchez discloses a logical device for handling dynamic attributes in a static directory comprising:
>
> a set of attribute declarations [list of attributes] containing at least one declaration for an attribute (see [0050]);
>
> at least one Real-time Attribute Processor (RTAP) [persistent data manager 81] configured to determine a value for an attribute (see [0044] and [0048]);
>
> an RTAP selector configured to select and invoke an RTAP according to a predetermined selection schema (see [0030], lines 7-15); and
>
> a directory attribute processor configured to parse requests for access to directory attribute values, to detect requests for attributes declared in said attribute declarations, to operate said RTAP selector to invoke a corresponding RTAP (see [0056]), to receive an attribute value determined by said invoked RTAP, and to return said attribute value to a requester [populating the object] (see [0062]).
>
> However, Sanchez fails to explicitly disclose the wherein the attributes are ynamic. Boreham discloses an X.500 (or LDAP) style directory

> service (see abstract and [0014]), including the further limitation of dynamic attributes (see [0226]) in order to process real-time data.
>
> It would have been obvious to one of ordinary skill in the art at the time of the invention to utilize dynamic attributes as disclosed by Boreham with the logical device of Sanchez. One would have been motivated to so in order to improve the efficiency when processing real-time data."

Applicant respectfully disagrees. Applicant's invention is directed towards dynamic handling of "real-time attributes" where the storage of the real-time data in an LDAP directory would be problematic, such as storing streaming stock ticker information in an LDAP directory (emphasis added to original disclosure text):

> *Applicant:*
>
> [0046] Our **Real-time** Attribute Processor ("RTAP") functionally cooperates with directory servers to handle **requests for dynamic attributes which would otherwise present a real-time processing challenge** to the directory server due to the server's dependence on the data normally being static in nature. Special schema syntax identifiers provided by the invention are used to identify attributes which are to be handled as **real-time attributes**, and **which are not to be stored directly in the directory, but whose values are resolved at the time a read request is made for those attributes.** This approach eliminates the need to store the dynamic information in the directory, and allows user-supplied modules to perform the resolution of the dynamic attributes in a real-time manner, including not only retrieving a value from a dynamic data source (e.g. a stock ticker), but optionally performing calculations or manipulations on the data as well (e.g. calculating a price-to-earnings ratio, watermarking an image, etc.).
>
> . . .
>
> [0049] We have developed an enhanced directory service framework whereby user- specified **attributes are never stored statically in the directory**, but instead are **resolved** in real-time by launching user implementations (e.g. a user-supplied program, script, or library function). Such real-time attributes are denoted using our special schema syntax marking, as described in the following paragraphs.

Thus, Applicant's invention, as disclosed and claimed, is patentably distinguished over Sanchez in view of Boreham at least in these ways:

(1) it handles requests for data which is *real-time* in nature, thus the nature of the data defies storage in static memory, and is inefficient to update in real-time in static memory such as the directory structure of an X.500 or LDAP directory; and

(2) it *dynamically* handles requests for information from a directory where that information is not actually stored in the directory (making it appear to the requester that the data *is* stored in the directory);

(3) it not only retrieves or looks-up a value dynamically, but *resolves* a real-time value from a real-time source; and

(4) it resolves real-time attribute data which is accessed from *outside* the static memory structure of the directory.

<u>(1) Real-Time Data is not the same as Dynamically Linked Data.</u> Applicant has described and claimed handling of data which is real-time in nature. "Real-time" is a term which is well-known in the computing art to mean processing and/or data which is time critical in nature, subject to change continuously:

> *<u>Dictionary:</u>*
>
> **real time** Immediate response by a computer system. The term is used to describe a number of different computer features. For example, real-time operating systems are systems that respond to in put immediately. They are used tor such tasks as navigation, in which the computer must react to a steady flow of new information without interruption. Most general purpose operating system are not real-time because they can take a few seconds, or even minutes to react.
>
> *Real time* can also refer to event simulated by a computer at the same speed that they would occur in real life. In graphics animation, for example, a real-time program would display objects moving across the screen at the same speed that they would actually move.
>
> => See also ISOCHRONOUS; OPERATING SYSTEM; OS/9
>
> *(Source: Random House Webster's "Computer & Internet Dictionary", Third Edition, by Philip E. Margolis, pg. 470)*

Applicant has referred to the attributes which the invention handles as "real-time attributes", and thus the claims have been amended to clarify and distinguish from "dynamic attributes" because the nature of Boreham's "dynamic attribute" is not data which is changing in real-time, but data which is likely to be changed in bulk.

As stated in the Office Action, Sanchez does not teach dynamic attributes.

Boreham discloses a method of handling their dynamic data using pointers to the dynamic data (e.g. a "look up" mechanism). Reference to any "dynamic attributes" only occurs twice in Boreham's disclosure, once in the portion relied upon for the Examiner's reasoning (para. 0226), and once earlier in para. 0208 emphasis added by Applicant):

> *Boreham:*
>
> "[0009] Also shown in FIG. 4 are entries. **Every entry in an X.500 Directory Information Tree (DIT)** is a collection of attributes, each attribute composed of a type element and one or more value elements. Because it was designed to accommodate all types of directories, in case of the X.500, the DAP has become too general to be easily configured to work with specialized applications. These reasons have resulted in a limited user acceptance of X.500. Each piece of information that describes some aspect of an entry is called an attribute. **An attribute comprises an attribute type and one or more attribute values**. An example of an attribute type might be `telephone number` and an example of a telephone number attribute value might be `+91 861 324 251`.
>
> . . .
>
> "[0207] As an example, suppose a directory contains thousands of entries all of which share a common attribute for fax number, **facsimileTelephoneNumber**. Traditionally, in order to change the fax number, a client application would have to update each entry individually. **This is a large administrative task, and sometimes there could be a risk of not updating all entries**.
>
> "[0208] CoS enables the user to generate the attribute **facsimileTelephoneNumber** dynamically. **The attribute facsimileTelephoneNumber is stored in one location in memory or in storage, and each entry points to that location to give a value to the**

> **entry's fax number attribute.** To a client application, the CoS attributes appear just like all other attributes of an entry, despite that CoS attributes are not stored with the entries themselves.
>
> . . .
>
> "Types of Classes of Service
>
> "[0226] Classes of service definitions (that is, all the information, save the template entries, needed in order to generate attribute values defined by a CoS) are stored in LDAP Subentries, **which can be located anywhere in the DIT**. A directory server user may use different types of CoS depending on how the user wishes to **generate the values of dynamic attributes**. There are three types of CoS: (a) Classic CoS; (b) Pointer CoS; and (c) Indirect CoS. Corresponding to these different types of CoS, CoS definition entries have objectclass cosClassicDefinition, cosPointerDefinition or cosIndirectDefinition."

As such, Boreham's "dynamic attribute" values are stored in "memory or storage" of the LDAP DIT, and are pointed to by pointers. This pointer-based approached could be referred to as a "look up" approach or a "dynamic link" approach. However, the values themselves are still stored in static memory, presumably within the LDAP DIT, as disclosed. While this approach relieves certain burdens on bulk updates of stored attribute values (e.g. bulk telephone number updates), it is not directed towards dynamically resolving values from external real-time source – there is no access of an external real-time source, nor is there any conversion of the data to match an LDAP return value format.

(2) Dynamic Handling is not Taught by Sanchez in view of Boreham. By "dynamic" handling and "dynamic resolving", Applicant is referring to, and claiming, a *variable process* which is invoked at the time of a request. The process varies according to the request by selecting which real-time attribute processor to invoke. The term "dynamic" is used in its conventional sense:

> *Dictionary:*
>
> **dynamic** Refers to actions that take place at the moment they are needed rather than in advance. For example, many programs perform *dynamic memory allocation*, which means that they do not reserve

> memory ahead of time but seize sections of memory when needed. In general, such programs require less memory, although they may run a little more slowly.
> The opposite of dynamic is *static.*
> ==> See also DYNAMIC VARIABLE; STATIC VARIABLE
> *(Source: Random House Webster's "Computer & Internet Dictionary", Third Edition, by Philip E. Margolis, pg. 182)*

Boreham's "dynamic attributes" are stored in memory or storage, and are pointed to by pointers. In other words, as exemplified by this definition of "dynamic", Boreham's memory to store their dynamic attributes is static and reserved in advance. Thus, their *process* or *action* of retrieving their values is not dynamic, but the values stored at the locations pointed to may change, making the *attributes* changeable (e.g. "dynamic attributes" not "dynamic attribute retrieving process").

<u>(3) Resolving a Value is not the Same as Looking Up a Value.</u> Applicant has disclosed, and claimed, that their real-time attribute values are *resolved* not just retrieved or looked up, because the data is not necessarily in an directory-compatible format having come from an external source (emphasis added to Applicant's original disclosure text):

> <u>Applicant:</u>
> [0046] Our Real-time Attribute Processor ("RTAP") functionally cooperates with directory servers to handle requests for dynamic attributes which would otherwise present a real-time processing challenge to the directory server due to the server's dependence on the data normally being static in nature. Special schema syntax identifiers provided by the invention are used to identify attributes which are to be handled as real-time attributes, and which are not to be stored directly in the directory, but whose values are resolved at the time a read request is made for those attributes. This approach eliminates the need to store the dynamic information in the directory, and allows user-supplied modules to perform the resolution of the dynamic attributes in a real-time manner, **<u>including not only retrieving a value from a dynamic data source (e.g. a stock ticker), but optionally performing calculations or manipulations on the data as well (e.g. calculating a</u>**

price-to-earnings ratio, watermarking an image, etc.). One preferred embodiment of the invention adheres to LDAP standards and protocols.

. . .

[0051] This example declaration for a dynamic attribute follows conventional BNF form, and declares a unique and previously-registered OID for the example attribute "currentTemp" to be **a value which is not statically stored in the directory, but which is to be resolved in real-time by a user-supplied process, module, method, or function.**

. . .

[0053] When a client application searches the directory for an entry that has a dynamic attribute type according to its OID, the directory server **invokes a function to resolve that attribute's current, real-time value.** According to our preferred embodiment, this function may be a portion of electronic logic, a software module, class, method, or other entity, preferably external to the directory server such that it may be supplied by the vendor of the directory server, or by the owner/operator of the directory server (e.g. customer-supplied). **By avoiding updating the value of the attribute when no client is requesting its value, our invention avoids unnecessary use of processor and communications bandwidth.** By updating the value of the attribute "on demand" in response to a request for the value, our invention provides real-time data rather than old or stale data.

. . .

[0056] During directory server processing for a search request, any requested entry that includes dynamic data is iterated over, and the data collected for each dynamic attribute through **resolving the values with the related external functions.** Following the "currentTemp" example above, a client would receive a reply such as:

```
City=Austin, state=Texas,c=US
Objectclass:city
CurrentTemp:82F
Population:602000
```

[0057] In this example, the values "Austin", "Texas", "US", and "602000 of the attributes city name, state, country, and population are stored and retrieved from the LDAP directory, as these items are relatively static in value. However, **the value "82F" of the real-time attribute**

> **CurrentTemp is not stored in the LDAP, and is resolved in real-time** by a function such as currentTemp.so. The function currentTemp.so in our prototype actually accesses real-time data from the online resource "weather.com", parses the data, and returns the value "82F" to the LDAP directory server, which then combines that information with the other static information from the LDAP directory, and passes it back to the requesting client.
>
> [0058] **It is important to note that the value "82F" was never actually stored in or retrieved from the LDAP directory.** In this manner, the handling of dynamic data is transparent to any requesting client, thereby providing a powerful extension to the directory server protocol to allow reading of dynamic data without changing or extending the protocol itself (e.g. client applications are backwards compatible to our improved directory server).

Thus, Applicant has claimed the action "resolve" to refer to a process of determining a requested value not by just looking up (e.g. using pointers) a value within the LDAP data structure, but by invoking a process external to the LDAP which obtains the value and converts it appropriately to satisfy the request. The term "resolve" is used conventionally in this manner:

> *Dictionary:*
>
> resolve
>
> . . .
>
> 4. To change or convert
>
> . . .
>
> *Source: The American Heritage® Dictionary of the English Language, Fourth Edition. Houghton Mifflin Company, 2004. 26 Jul. 2007. <Dictionary.com http://dictionary.reference.com/browse/resolve>.*

Applicant has amended independent Claims 1, 8 and 14 to specify that the real-time value is "dynamically resolved" through accessing from a source external to the directory, and through converting the directory-external value to a format compatible with a return parameter of a directory access request (i.e. temperature access from web site example).

(4) Accessing Data from Outside the LDAP Structure is Not Taught. As described (paras. 0046 - 0057 quoted above) and claimed by Applicant, the real-time values handled by the invention come from *outside* the LDAP directory from a real-time source (e.g. a stock ticker data feed, a weather web site, etc.). However, Sanchez in view of Boreham is silent regarding this aspect of the claims.

Summary. Applicant submits that it would have been an unreasonable leap in reason for one ordinarily skilled in the art at the time of the invention who had knowledge Sanchez and Boreham disclosures to envision modifying Boreham's pointer structure to dynamic values in memory by replacing them with directory-external functions which resolve values from non-directory sources. There are at least two major differences between the pointer approach and the external function/external source approach which would not have reasonably occurred to one of ordinary skill in the art:

	Sanchez/Boreham	Applicant
Source of data:	memory or storage within LDAP DIT	external to LDAP structure
Format of data:	LDAP-compatible	LDAP-unaware (likely needs formatting and conversion to satisfy request)
Mechanism to locate and find data:	pointers (e.g. "look up" process)	process or function external to LDAP, intelligent enough to locate data at its source and to convert it appropriately

For these reasons, Applicant requests allowance of Claim 1.

Rejection of Dependent Claim 2. In the Office Action starting on Page 4, Claim 2 was rejected reasoning:

> Examiner:
>
> ". . . the combination of Sanchez and Boreham (hereafter Sanchez/Boreham) discloses the logical device as set forth in claim 1 wherein said directory attribute processor is further adapted to suppress storage of said attribute value in a directory [use standard attributes] (Sanchez: see [0029]-[0031])."

Applicant respectfully disagrees. Sanchez specifically specifies in the paragraphs cited by the Examiner that their data values are stored in the LDAP data repository (emphasis added by Applicant):

> *Sanchez:*
>
> "[0029] Since objects 48 are defined by their attributes, the persistent data manager 40 stores objects 48 **by storing persistent attributes of the objects 48 in the LDAP data repository 26**. Accordingly, when directed to store an object 48 in the LDAP data repository 26, the persistent data manager 40 **stores the persistent attributes in the LDAP data repository** 26. Likewise, when directed to read a stored object 48 from the LDAP data repository 26, the persistent data manager 40 **reads the persistent attributes from the LDAP data repository 26**.
>
> [0030] The persistent data manager 40 preferably stores the persistent attributes by mapping the persistent attributes to certain defined LDAP attributes. These LDAP attributes are preferably grouped within LDAP objects (not shown) that correspond to the objects 48. These LDAP objects and their attributes are defined within the LDAP data repository 26. In an embodiment of the present invention, the persistent data manager 40 is responsible for defining, in the LDAP data repository 26, most of the LDAP attributes and the LDAP objects that correspond to the objects 48. Some of the LDAP attributes may be standard LDAP attributes that already exist. These LDAP attributes and objects, referred to as persistent data schema, may be defined and created using known LDAP methods for creating LDAP attributes and objects.

> [0031] The persistent data manager 40 may map the persistent attributes of the objects 48 to the LDAP attributes utilizing a simple mapping methodology. For example, the created LDAP attributes may be defined to have the same names as the persistent attributes with a simple prefix (or suffix or other notation) indicating a source (e.g., an organization and an application) from which the persistent attributes originate. For example, a time stamp attribute of an object 48, named "CreatedTime" may map to a created LDAP attribute named "hpmxCreatedTime." The prefix "hpm.times." may indicate, for example, that the source organization is Hewlett-Packard Co..RTM. and the source application is the SCM 12 (also known as MX or MUXPlex). As noted above, the persistent data manager 40 may map some of the persistent attributes to existing standard LDAP attributes (e.g., commonName, description, ipHostNumber, host and uidNumber)."

Applicant respectfully submits that this portion of Sanchez' disclosure has been misinterpreted, possibly by improperly reading the Applicant's disclosure into Sanchez' disclosure, because there is no mention of Sanchez' system storing their "persistent attributes" anywhere but "in the LDAP data repository.

For this reason, Applicant requests allowance of Claim 2.

Rejection of Dependent Claims 3 - 7. In the Office Action, starting on Page 4, the reasoning for rejecting were provided. Applicant respectfully disagrees with the reasoning and conclusions as stated by the Examiner.

Claims 3 - 7 depend on Claim 1. All of the elements, steps, and limitations of Claim 1 are not taught by Sanchez in view of Boreham, as discussed in the foregoing paragraphs. For this reason, Applicant requests allowance of Claims 3 - 7.

Rejection of Independent Method Claim 8. In the Office Action, starting on Page 5, reasons for rejecting method Claim 8 were provided:

> Examiner:
>
> "Referring to claim 8, Sanchez discloses a method for handling dynamic attributes in a static director/ server comprising:
>
> providing at least one declaration for an attribute in association with a set of directory attribute declarations [list of attributes] (see [0050]);
>
> parsing requests for access to directory attribute values to detect requests for attributes declared in said attribute declarations (see [0056]);
>
> invoking at least one Real-time Attribute Processor (RTAP) selected according to a predetermined selection schema, said RTAP being configured to determine a value for an attribute declared as said set of attribute declarations, said dynamic value being unavailable from said static directory (see [0056]); and
>
> returning to a requester an attribute value determined by said invoked RTAP [populating the object] (see [0062]).
>
> "However, Sanchez fails to explicitly disclose the wherein the attributes are dynamic. Boreham discloses an X.500 (or LDAP) style directory service (see abstract and [0014]), including the further limitation of dynamic attributes (see [0226]) in order to process real-time data.
>
> "It would have been obvious to one of ordinary skill in the art at the time of the invention to utilize dynamic attributes as disclosed by Boreham with the logical device of Sanchez. One would have been motivated to so in order to improve the efficiency when processing real-time data."

Applicant agrees with the Examiner's statement that Sanchez fails to teach attributes which are dynamic, but respectfully disagrees with the Examiner's statements regarding Sanchez' teachings, Boreham's teachings, and the obviousness to combine and modify those teachings to meet the steps, elements, and limitations of Applicant's claims.

Claim 8 is directed toward a method embodiment of the invention. The aspects of Claim 1, which are untaught by Sanchez or Boreham, are present in Claim 8, including:

(a) handling of *real-time* data, not just data which needs to be changed in bulk (Boreham);

(b) *dynamic* handling of data requests where the handling process is dynamic, not just a static look-up (Boreham) or mapping (Sanchez) process;

(c) *resolving* of data (including parsing and formatting as required), not just retrieving of directory-compatible data; and

(d) accessing of real-time data from *outside* the directory structure, not inside it.

The points illustrated in the foregoing paragraphs regarding the rejection of independent Claim 1 are equally applicable and relevant to the rejection of independent Claim 8. It would have been unreasonable, as pointed out in those paragraphs, for one of ordinary skill in the art to modify Sanchez and Boreham in so many ways away from their actual teachings to meet the limitations of Claim 8.

For these reasons, Applicant requests allowance of Claim 8.

Rejections of Dependent Claims 9 - 12. In the Office Action, reasons for rejecting Claims 9 - 12 were provided starting on Page 6. These claims dependent from Claim 8. The method embodiments of claims 9 - 12 correspond to the system embodiment claims 3 - 6, respectively.

The points illustrated in the foregoing paragraphs regarding the rejections of dependent Claims 3 - 6 are equally applicable and relevant to the rejection of dependent Claims 9 - 12. It would have been unreasonable, as pointed out in those paragraphs, for one of ordinary skill in the art to modify Sanchez and Boreham in so many ways away from their actual teachings to meet the limitations of Claims 9 - 12.

For these reasons, Applicant requests allowance of Claims 9 - 12.

Rejection of Dependent Claim 13. In the Office Action starting on Page 7, the Examiner provided the reasons for rejection of Claim 13:

> *Examiner:*
> "Referring to claim 13, Sanchez/Boreham discloses the method as set forth in claim 8 wherein said step of returning to-a requester an attribute value comprising returning said value [populating the object] according to a Lightweight Directory Access Protocol (Sanchez: see [0062])."

Applicant respectfully disagrees. Claim 13 depends from Claim 8, and thus the shortcomings of the proposed Sanchez-Boreham combination apply to Claim 13, as well.

However, in Claim 13, Applicant has specified that the returned value to the requester is returned according to the LDAP protocol. As described in the foregoing paragraphs regarding the resolving of the value from its original non-LDAP directory-external source, this value has necessarily been parsed, converted, or formatted from its non-LDAP original format. Thus, Claim 13 is not simply specifying a protocol, which is well known, but by virtue of its dependence from Claim 8, it is further specifying a conversion process, the product of which is LDAP compatible.

For these reasons, Applicant requests allowance of Claim 13.

Rejections of Claims 14 - 19. In the Office Action, starting on Page 17, the Examiner has provided reasons for rejection of computer readable medium claims 14 - 19. Applicant respectfully disagrees with the Examiner's statement of reasons and conclusions regarding the teachings of Sanchez and Boreham, and regarding what would have been obvious to do by one of ordinary skill in the art at the time of the invention.

Claim 14 is independent, and claims 15 - 19 dependent from Claim 14. Furthermore, Claims 14 - 19 are directed towards computer readable medium embodiments which correspond to the method embodiment claims 8 - 13.

As such, the reasons why Sanchez in view of Boreham fails to teach all of the claim elements, steps, and limitations, and why it would have been unreasonable for an ordinarily skilled person in the art to make so many changes without teaching or suggestion by Sanchez or

Boreham as set forth in the foregoing paragraphs are equally applicable to the rejections of Claims 14 - 19.

For these reasons, Applicant requests allowance of Claims 14 - 19.

Respectfully,

/ Robert Frantz /

Robert H. Frantz, Reg. No. 42,553
Agent for Applicant
Tel: (405) 812-5613
Franklin Gray Patents, LLC